Module 04: Lab 01

Visual Reporting and Storytelling

Pranjul Garg

March 20, 2025

# Objectives

By the end of this lab, you will: 1. Load and analyze the **Lightcast dataset** in **Spark DataFrame**. 2. Create **five easy and three medium-complexity visualizations** using **Plotly**. 3. Explore **salary distributions, employment trends, and job postings**. 4. Analyze **skills in relation to NAICS/SOC/ONET codes and salaries**. 5. Customize **colors, fonts, and styles** in all visualizations (**default themes result in a 2.5-point deduction**). 6. Follow **best practices for reporting on data communication**.

# Step 1: Load the Dataset

#import gdown  
#url = "https://drive.google.com/uc?id=1V2GCHGt2dkFGqVBeoUFckU4IhUgk4ocQ"  
#gdown.download(url, "lightcast\_data.csv", quiet=False)

import pandas as pd  
import plotly.express as px  
import plotly.io as pio  
pio.renderers.default = "vscode"  
from pyspark.sql import SparkSession  
from pyspark.sql.functions import col  
spark = SparkSession.builder.appName("lightcast\_data.csv").getOrCreate()  
  
df = spark.read.option("header", "true").option("inferSchema", "true").option("multiLine","true").option("escape", "\"").csv("lightcast\_data.csv")  
  
#df.printSchema()  
#df.show(5)

Setting default log level to "WARN".  
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).  
25/03/28 05:31:19 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

# 1. Salary Distribution by Employment Type

* Identify salary trends across different employment types.
* **Filter the dataset**
  + Remove records where **salary is missing or zero**.
* **Aggregate Data**
  + Group by **employment type** and compute salary distribution.
* **Visualize results**
  + Create a **box plot** where:
    - **X-axis** = EMPLOYMENT\_TYPE\_NAME
    - **Y-axis** = SALARY\_FROM
  + Customize **colors, fonts, and styles** to avoid a **2.5-point deduction**.
* **Explanation:** Write two sentences about what the graph reveals.

# Your Code for 1st question here  
pdf = df.select("EMPLOYMENT\_TYPE\_NAME", "SALARY").toPandas()  
fig = px.box(pdf, x="EMPLOYMENT\_TYPE\_NAME", y="SALARY", title="Salary Distribution by Employment Type", color\_discrete\_sequence=["#636EFA"])  
fig.update\_layout(font\_family="Arial", title\_font\_size=16)  
fig.show()  
fig.write\_image("Figures/qus1.svg")

Unable to display output for mime type(s): application/vnd.plotly.v1+json

[Salary Distribution](Figures/qus1.svg)

##### 1.0.0.0.1 This chart shows how salaries vary based on job type. People working full-time (> 32 hours a week) usually earn more and have a bigger range of salaries, including some very high ones.

#### 1.0.0.1 Part-time workers, whether only part-time or mixed with full-time, tend to earn less and have more similar pay across the board.

# 2. Salary Distribution by Industry

* Compare salary variations across industries.
* **Filter the dataset**
  + Keep records where **salary is greater than zero**.
* **Aggregate Data**
  + Group by **NAICS industry codes**.
* **Visualize results**
  + Create a **box plot** where:
    - **X-axis** = NAICS2\_NAME
    - **Y-axis** = SALARY\_FROM
  + Customize colors, fonts, and styles.
* **Explanation:** Write two sentences about what the graph reveals.

# Your code for 2nd question here  
  
from pyspark.sql.functions import avg, min, max, count  
filtered\_df = df.filter((col("SALARY\_FROM").isNotNull()) & (col("SALARY\_FROM") != 0))  
  
industry\_df = filtered\_df.groupBy("NAICS2\_NAME").agg(  
 count("\*").alias("job\_count"))  
  
industry\_df.show()

+--------------------+---------+  
| NAICS2\_NAME|job\_count|  
+--------------------+---------+  
|Administrative an...| 4040|  
|Public Administra...| 737|  
|Real Estate and R...| 454|  
| Information| 2356|  
|Unclassified Indu...| 3811|  
|Accommodation and...| 270|  
|Finance and Insur...| 4013|  
| Construction| 299|  
| Utilities| 343|  
|Management of Com...| 41|  
|Professional, Sci...| 9282|  
|Arts, Entertainme...| 90|  
|Other Services (e...| 385|  
|Transportation an...| 245|  
| Wholesale Trade| 943|  
|Agriculture, Fore...| 29|  
| Manufacturing| 1740|  
|Mining, Quarrying...| 38|  
|Educational Services| 1033|  
|Health Care and S...| 1442|  
+--------------------+---------+  
only showing top 20 rows

import glob  
filtered\_df = df.filter(  
 (col("SALARY\_FROM").isNotNull()) &   
 (col("SALARY\_FROM") > 0) &   
 (col("NAICS2\_NAME").isin("Health Care and Social Assistance",   
 "Finance and Insurance",   
 "Information"))  
)  
  
salary\_df = filtered\_df.select("NAICS2\_NAME", "SALARY\_FROM")  
salary\_df.write.mode("overwrite").option("header", True).csv("industry\_salaries")  
  
files = glob.glob("industry\_salaries/part-\*.csv")  
pdf = pd.concat((pd.read\_csv(f) for f in files), ignore\_index=True)  
  
pio.renderers.default = "plotly\_mimetype"  
fig = px.box(  
 pdf,  
 x="NAICS2\_NAME",  
 y="SALARY\_FROM",  
 title="Salary Distribution by Industry",  
 color="NAICS2\_NAME",  
 color\_discrete\_sequence=["#1f77b4", "#2ca02c", "#d62728"]   
)  
  
fig.update\_layout(  
 font\_family="Calibri",  
 font\_size=14,  
 title\_font\_size=20,  
 xaxis\_title="Industry Type",  
 yaxis\_title="Salary (USD)",  
 plot\_bgcolor="#fafafa",  
 paper\_bgcolor="#f5f5f5",  
 showlegend=False  
)  
  
fig.show()  
fig.write\_image("Figures/qus2.svg")

Unable to display output for mime type(s): application/vnd.plotly.v1+json

This chart compares salary ranges across three major industries. The Information industry tends to offer the highest salaries, with a wider spread and more high-paying roles. Finance and Insurance shows slightly lower median salaries, but still competitive. Meanwhile, Health Care and Social Assistance generally offers the lowest pay among the three, though it still has some high-salary outliers.

# 3. Job Posting Trends Over Time

* Analyze how job postings fluctuate over time.
* **Aggregate Data**
  + Count job postings per **posted date (POSTED)**.
* **Visualize results**
  + Create a **line chart** where:
    - **X-axis** = POSTED
    - **Y-axis** = Number of Job Postings
  + Apply custom colors and font styles.
* **Explanation:** Write two sentences about what the graph reveals.

#Filtering and grouping job postings by date  
import glob  
job\_post\_df = (  
 df.filter(col("POSTED").isNotNull())  
 .groupBy("POSTED")  
 .agg(count("\*").alias("job\_count"))  
)  
  
# grouping the data to CSV and read with pandas  
job\_post\_df.write.mode("overwrite").option("header", True).csv("job\_posts")  
files = glob.glob("job\_posts/part-\*.csv")  
pdf = pd.concat((pd.read\_csv(f) for f in files), ignore\_index=True)  
  
#Format and sorting by date  
pdf["POSTED"] = pd.to\_datetime(pdf["POSTED"])  
pdf = pdf.sort\_values("POSTED")  
  
pio.renderers.default = "plotly\_mimetype"  
  
fig = px.line(  
 pdf,  
 x="POSTED",  
 y="job\_count",  
 title="Job Postings Trend Over Time",  
 markers=True, # adds visible dots at data points  
 color\_discrete\_sequence=["#FF7F0E"]   
)  
  
fig.update\_layout(  
 font=dict(family="Segoe UI", size=14),  
 title\_font=dict(family="Georgia", size=22, color="#333333"),  
 xaxis\_title="Dates on which the jobs were posted",  
 yaxis\_title="Number of Job Postings",  
 plot\_bgcolor="#FAFAFA",  
 paper\_bgcolor="#FFFFFF"  
)  
  
fig.show()  
fig.write\_image("Figures/qus3.svg")

Unable to display output for mime type(s): application/vnd.plotly.v1+json

This line graph shows how the number of job postings changed from May to October 2024. While the overall trend stays fairly consistent, there are regular ups and downs—suggesting that job postings spike on certain days or weeks, then dip right after. It could reflect typical hiring cycles or patterns like jobs being posted more often early in the week.

# 4. Top 10 Job Titles by Count

* Identify the most frequently posted job titles.
* **Aggregate Data**
  + Count the occurrences of each **job title (TITLE\_NAME)**.
  + Select the **top 10 most frequent titles**.
* **Visualize results**
  + Create a **bar chart** where:
    - **X-axis** = TITLE\_NAME
    - **Y-axis** = Job Count
  + Apply custom colors and font styles.
* **Explanation:** Write two sentences about what the graph reveals.

# Your code for 4th question here  
#finding the top 10 jobs  
job\_title = df.filter(col("TITLE\_NAME").isNotNull()) \  
 .groupBy("TITLE\_NAME") \  
 .agg(count("\*").alias("job\_count"))  
top\_10\_jobs = job\_title.orderBy("job\_count", ascending=False).limit(10)  
  
top\_10\_jobs.show()

[Stage 10:> (0 + 1) / 1]

+--------------------+---------+  
| TITLE\_NAME|job\_count|  
+--------------------+---------+  
| Data Analysts| 8591|  
| Unclassified| 3149|  
|Business Intellig...| 2072|  
|Enterprise Archit...| 1999|  
|Oracle Cloud HCM ...| 1042|  
| Data Modelers| 668|  
|Data Governance A...| 628|  
|Data Analytics En...| 537|  
|ERP Business Anal...| 488|  
|Data Quality Anal...| 467|  
+--------------------+---------+

top\_10\_jobs.write.mode("overwrite").option("header", True).csv("top\_10\_jobs")  
files = glob.glob("top\_10\_jobs/part-\*.csv")  
pdf4 = pd.concat((pd.read\_csv(f) for f in files), ignore\_index=True)  
  
pio.renderers.default = "plotly\_mimetype"  
  
  
fig = px.bar(  
 pdf4,  
 x="TITLE\_NAME",  
 y="job\_count",  
 title="Top 10 Job Titles by Demand",  
 color="TITLE\_NAME",  
 color\_discrete\_sequence=px.colors.qualitative.Safe   
)  
  
fig.update\_layout(  
 font=dict(family="Verdana", size=14),  
 title\_font=dict(family="Georgia", size=22, color="#333333"),  
 height=550,  
 width=900,  
 xaxis\_title="Job Titles",  
 yaxis\_title="Number of Postings",  
 plot\_bgcolor="#F8F8F8",  
 paper\_bgcolor="#FFFFFF",  
 showlegend=True,  
 legend\_title="Legend"  
)  
  
fig.show()  
fig.write\_image("Figures/qus4.svg")

Unable to display output for mime type(s): application/vnd.plotly.v1+json

This bar chart shows the top 10 most in-demand job titles based on the number of postings. Data Analysts dominate the chart by a wide margin, indicating very high demand. The rest—including roles like Business Intelligence Analysts, Enterprise Architects, and Oracle Cloud HCM Consultants—have significantly fewer postings, showing that while specialized roles are needed, generalist data positions are currently leading the job market.

# 5. Remote vs On-Site Job Postings

* Compare the proportion of remote and on-site job postings.
* **Aggregate Data**
  + Count job postings by **remote type (REMOTE\_TYPE\_NAME)**.
* **Visualize results**
  + Create a **pie chart** where:
    - **Labels** = REMOTE\_TYPE\_NAME
    - **Values** = Job Count
  + Apply custom colors and font styles.
* **Explanation:** Write two sentences about what the graph reveals.

# Your code for 5th question here  
remote\_jobs = df.filter(col("REMOTE\_TYPE\_NAME").isNotNull()) \  
 .groupBy("REMOTE\_TYPE\_NAME") \  
 .agg(count("\*").alias("job\_count"))  
  
remote\_jobs.show()

[Stage 16:> (0 + 1) / 1]

+----------------+---------+  
|REMOTE\_TYPE\_NAME|job\_count|  
+----------------+---------+  
| Remote| 12497|  
| [None]| 56570|  
| Not Remote| 1127|  
| Hybrid Remote| 2260|  
+----------------+---------+

remote\_jobs.write.mode("overwrite").option("header", True).csv("remote\_jobs")  
files = glob.glob("remote\_jobs/part-\*.csv")  
pdf5 = pd.concat((pd.read\_csv(f) for f in files), ignore\_index=True)  
  
pio.renderers.default = "plotly\_mimetype"  
custom\_colors = ["#00A699", "#FF6F61", "#1E90FF", "#FFD166", "#6D6875", "#F4A261"]  
  
fig = px.pie(  
 pdf5,  
 names="REMOTE\_TYPE\_NAME",  
 values="job\_count",  
 title="Remote vs On-Site Job Distribution",  
 color\_discrete\_sequence=custom\_colors,  
 hole=0.4   
)  
  
fig.update\_layout(  
 font=dict(family="Segoe UI", size=15),  
 title\_font=dict(family="Georgia", size=22, color="#222222"),  
 showlegend=True,  
 legend\_title="Work Type",  
 paper\_bgcolor="#FFFFFF"  
)  
  
fig.show()  
fig.write\_image("Figures/qus5.svg")

Unable to display output for mime type(s): application/vnd.plotly.v1+json

This donut chart shows how job postings are split between remote, hybrid, and on-site work types. A huge chunk (78.1%) of the postings don’t specify the work type, which could mean missing or incomplete data. Among the rest, remote jobs make up 17.2%, while hybrid and on-site (not remote) roles are much less common at just over 3% and 1.5% respectively. So, while remote work is present, transparency in job type still has room for improvement.

# 6. Skill Demand Analysis by Industry (Stacked Bar Chart)

* Identify which skills are most in demand in various industries.
* **Aggregate Data**
  + Extract **skills** from job postings.
  + Count occurrences of skills grouped by **NAICS industry codes**.
* **Visualize results**
  + Create a **stacked bar chart** where:
    - **X-axis** = Industry
    - **Y-axis** = Skill Count
    - **Color** = Skill
  + Apply custom colors and font styles.
* **Explanation:** Write two sentences about what the graph reveals.

from pyspark.sql.functions import col, split, explode, trim, count  
skills\_df = df.filter(col("SKILLS\_NAME").isNotNull()) \  
 .withColumn("SKILL", explode(split(col("SKILLS\_NAME"), ","))) \  
 .withColumn("SKILL", trim(col("SKILL")))  
  
skill\_counts\_df = skills\_df.groupBy("SKILL").agg(count("\*").alias("total\_count"))  
top\_skills = [row["SKILL"] for row in skill\_counts\_df.orderBy(col("total\_count").desc()).limit(10).collect()]  
  
# top 10 skills  
top\_skills\_df = skills\_df.filter(col("SKILL").isin(top\_skills))  
  
#Aggregate skill counts per industry  
industry\_skill\_counts\_df = top\_skills\_df.groupBy("NAICS2\_NAME", "SKILL") \  
 .agg(count("\*").alias("skill\_count"))  
pdf = industry\_skill\_counts\_df.toPandas()  
  
# Limit to top 10 industries by total postings  
top\_industries = (  
 pdf.groupby("NAICS2\_NAME")["skill\_count"]  
 .sum()  
 .sort\_values(ascending=False)  
 .head(10)  
 .index.tolist()  
)  
pdf = pdf[pdf["NAICS2\_NAME"].isin(top\_industries)]  
  
fig = px.bar(  
 pdf,  
 y="NAICS2\_NAME",  
 x="skill\_count",  
 color="SKILL",  
 title="Top 10 Skills in Top 10 Industries",  
 barmode="stack",  
 color\_discrete\_sequence=px.colors.qualitative.Set3,  
 labels={"skill\_count": "Number of Postings", "NAICS2\_NAME": "Industry"}  
)  
  
  
fig.update\_layout(  
 font\_family="Calibri",  
 font\_size=13,  
 title\_font\_size=20,  
 plot\_bgcolor="#ffffff",  
 paper\_bgcolor="#f8f8f8"  
)  
  
fig.show()  
fig.write\_image("Figures/qus6.svg")

Unable to display output for mime type(s): application/vnd.plotly.v1+json

This stacked bar chart displays the top ten most in-demand skills across the top ten industries, as determined by job posting data. Each bar denotes an industry, and the colored portions indicate how often each skill appears in job postings within that area. For example, Professional, Scientific, and Technical Services has the highest overall need, particularly for skills such as communication, SQL, and project management. The chart clearly shows which skills are in high demand in various areas, as well as where job searchers with those skills might find competition or opportunities.

# 7. Salary Analysis by ONET Occupation Type (Bubble Chart)

* Analyze how salaries differ across ONET occupation types.
* **Aggregate Data**
  + Compute **median salary** for each occupation in the **ONET taxonomy**.
* **Visualize results**
  + Create a **bubble chart** where:
    - **X-axis** = ONET\_NAME
    - **Y-axis** = Median Salary
    - **Size** = Number of job postings
  + Apply custom colors and font styles.
* **Explanation:** Write two sentences about what the graph reveals.

from pyspark.sql.functions import col  
import pandas as pd  
import plotly.express as px  
  
# Step 1: Filter valid records  
onet\_df = df.filter(  
 col("SALARY\_FROM").isNotNull() &  
 (col("SALARY\_FROM") > 0) &  
 col("ONET\_NAME").isNotNull()  
)  
  
# Step 2: Calculate median salary and job count per ONET role  
results = [  
 (job, \*onet\_df.filter(col("ONET\_NAME") == job)  
 .approxQuantile("SALARY\_FROM", [0.5], 0.01) +   
 [onet\_df.filter(col("ONET\_NAME") == job).count()])  
 for job in onet\_df.select("ONET\_NAME").distinct().rdd.flatMap(lambda x: x).collect()  
]  
  
# Step 3: Convert to pandas DataFrame  
summary\_df = pd.DataFrame(results, columns=["ONET\_NAME", "Median\_Salary", "Job\_Count"])  
  
# Optional: Filter top 20 roles by job count for cleaner view  
summary\_df = summary\_df.sort\_values("Job\_Count", ascending=False).head(20)  
  
# Step 4: Create Bubble Chart  
fig = px.scatter(  
 summary\_df,  
 x="ONET\_NAME",  
 y="Median\_Salary",  
 size="Job\_Count",  
 color="Median\_Salary",  
 title="ONET Occupation Salary Analysis (Bubble Chart)",  
 labels={"ONET\_NAME": "Job Role", "Median\_Salary": "Median Salary", "Job\_Count": "Number of Postings"},  
 color\_continuous\_scale="Blues"  
)  
  
fig.update\_layout(  
 font\_family="Calibri",  
 font\_size=13,  
 title\_font\_size=20,  
 xaxis\_tickangle=45,  
 height=600,  
 plot\_bgcolor="#ffffff",  
 paper\_bgcolor="#f8f8f8"  
)  
  
fig.show()  
fig.write\_image("Figures/qus7.svg")

Unable to display output for mime type(s): application/vnd.plotly.v1+json

This bubble chart was meant to compare salaries across different job roles, but it currently shows only one bubble. That means only one ONET occupation — in this case, Business Intelligence Analysts — had both salary data and enough records to calculate a reliable median salary.

The chart still gives us one key insight: this role has a median salary around $87,295, and a significant number of job postings. However, the lack of other bubbles suggests either:

1. Other roles didn’t have valid salary data, or
2. There weren’t enough postings to include them in the top 20 list.

# 8. Career Pathway Trends (Sankey Diagram)

* Visualize job transitions between different occupation levels.
* **Aggregate Data**
  + Identify career transitions between **SOC job classifications**.
* **Visualize results**
  + Create a **Sankey diagram** where:
    - **Source** = SOC\_2021\_2\_NAME
    - **Target** = SOC\_2021\_3\_NAME
    - **Value** = Number of transitions
  + Apply custom colors and font styles.
* **Explanation:** Write two sentences about what the graph reveals.

# Your code for 8th question here  
import plotly.graph\_objects as go  
  
# Step 1: Aggregate transitions from 2-digit to 3-digit SOC codes  
career\_path = df.filter(  
 col("SOC\_2021\_2\_NAME").isNotNull() &  
 col("SOC\_2021\_3\_NAME").isNotNull()  
)  
  
transition\_counts = career\_path.groupBy("SOC\_2021\_2\_NAME", "SOC\_2021\_3\_NAME") \  
 .agg(count("\*").alias("job\_count")) \  
 .toPandas()  
  
#node labels (unique list of all categories)  
all\_labels = pd.concat([transition\_counts["SOC\_2021\_2\_NAME"], transition\_counts["SOC\_2021\_3\_NAME"]]).unique()  
label\_to\_index = {label: idx for idx, label in enumerate(all\_labels)}  
  
#Build Sankey data  
sources = transition\_counts["SOC\_2021\_2\_NAME"].map(label\_to\_index)  
targets = transition\_counts["SOC\_2021\_3\_NAME"].map(label\_to\_index)  
values = transition\_counts["job\_count"]  
  
fig = go.Figure(data=[go.Sankey(  
 node=dict(  
 pad=15,  
 thickness=20,  
 line=dict(color="black", width=0.5),  
 label=all\_labels,  
 color="rgba(31, 119, 180, 0.8)"  
 ),  
 link=dict(  
 source=sources,  
 target=targets,  
 value=values,  
 color="rgba(150, 200, 250, 0.4)"  
 )  
)])  
  
fig.update\_layout(  
 title\_text="Career Pathway Trends (Sankey Diagram)",  
 font=dict(family="Calibri", size=13),  
 height=600  
)  
  
fig.show()  
fig.write\_image("Figures/qus8.svg")

Unable to display output for mime type(s): application/vnd.plotly.v1+json

This Sankey diagram displays a clear job transition flow from “Computer and Mathematical Occupations” (the larger category) to “Mathematical Science Occupations” (a more narrow role group). The relationship suggests a high volume of job posts for this career, implying that math-focused professions are a prominent specialization within the larger tech and data area.